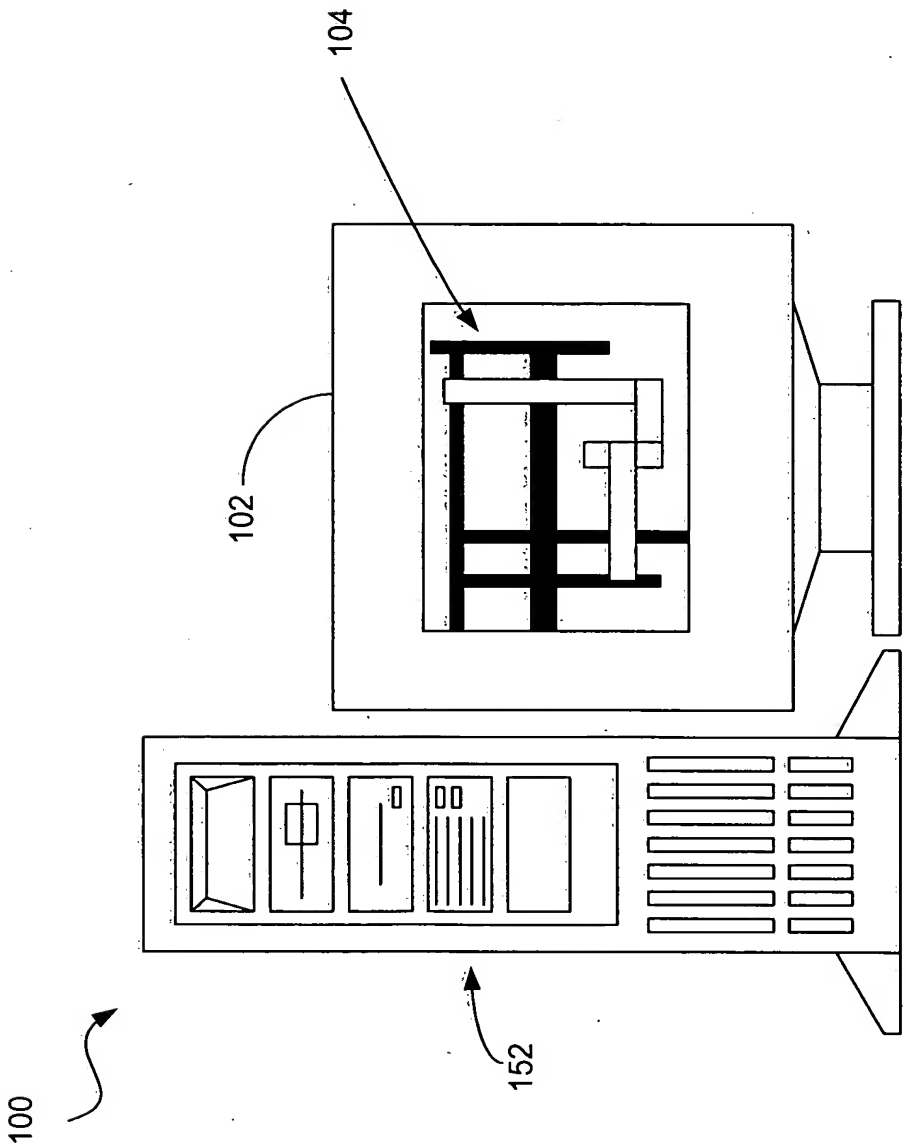
FIG. 1



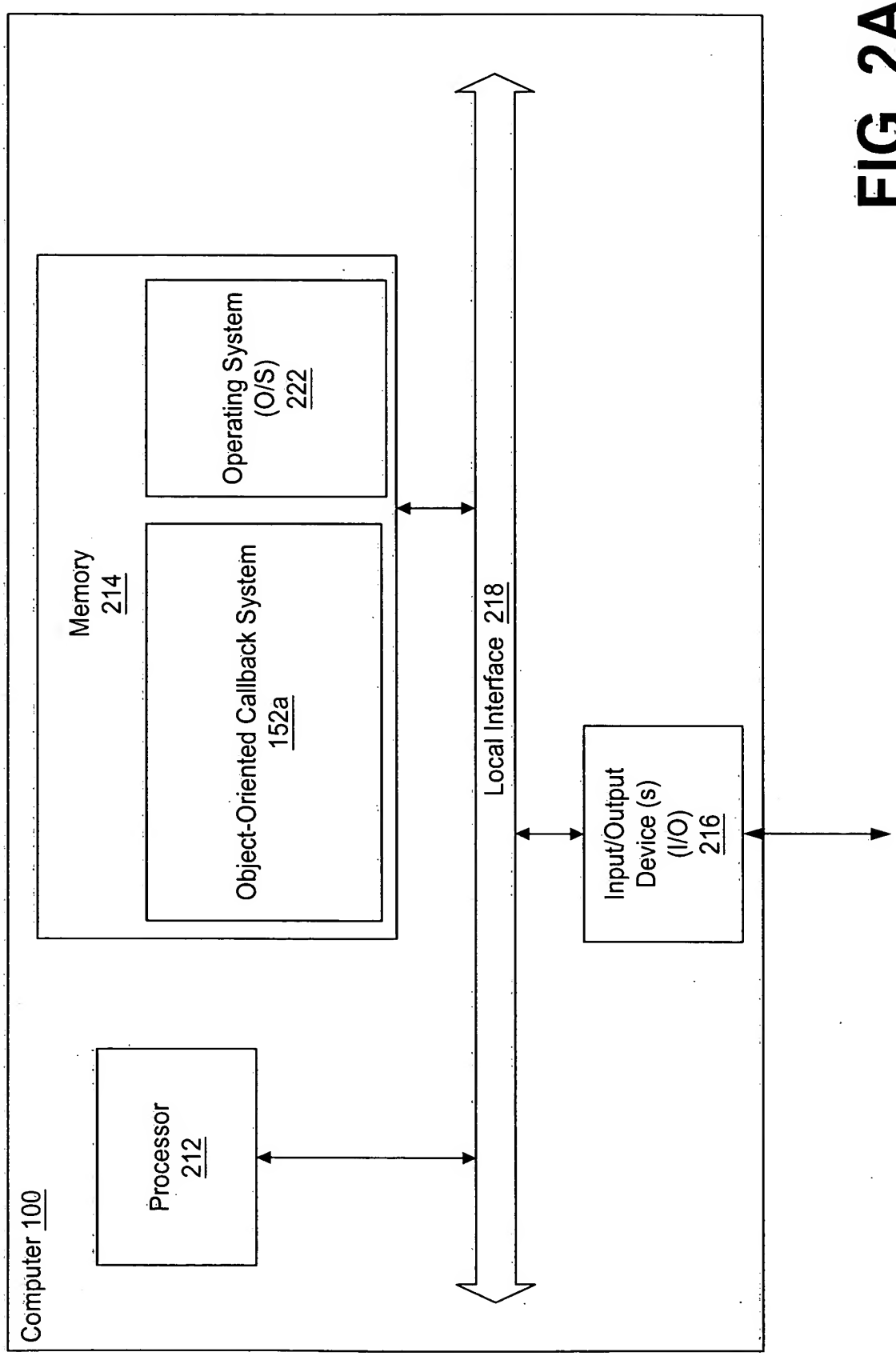

Computer 100
Processor 212
Memory 214
Operating System (O/S) 222
Object-Oriented Callback System 152a
Local Interface 218
Input/Output Device (s) (I/O) 216

FIG. 2A

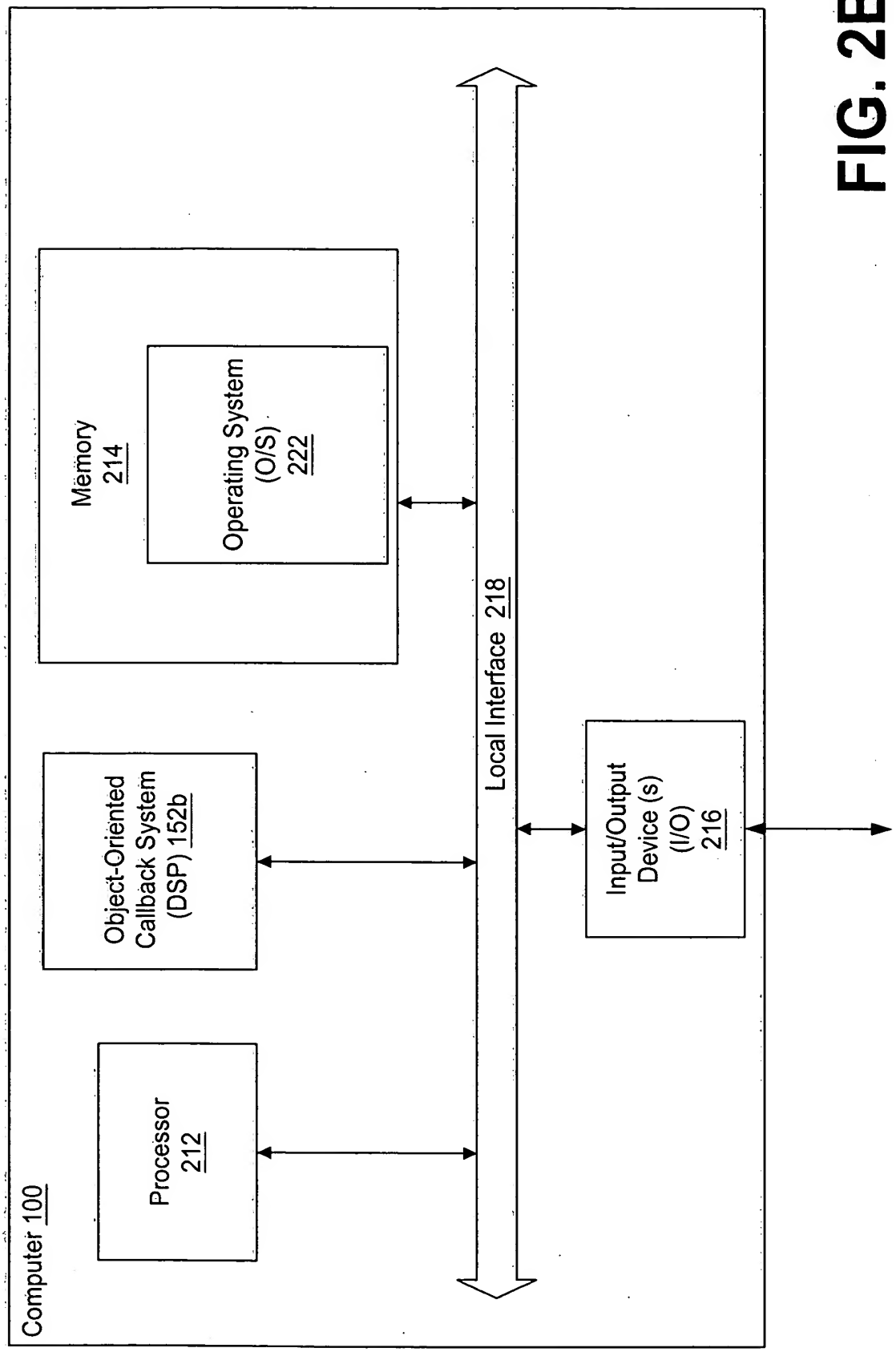

Computer 100
Processor 212
Object-Oriented Callback System (DSP) 152b
Memory 214
Operating System (O/S) 222
Local Interface 218
Input/Output Device (s) (I/O) 216

FIG. 2B

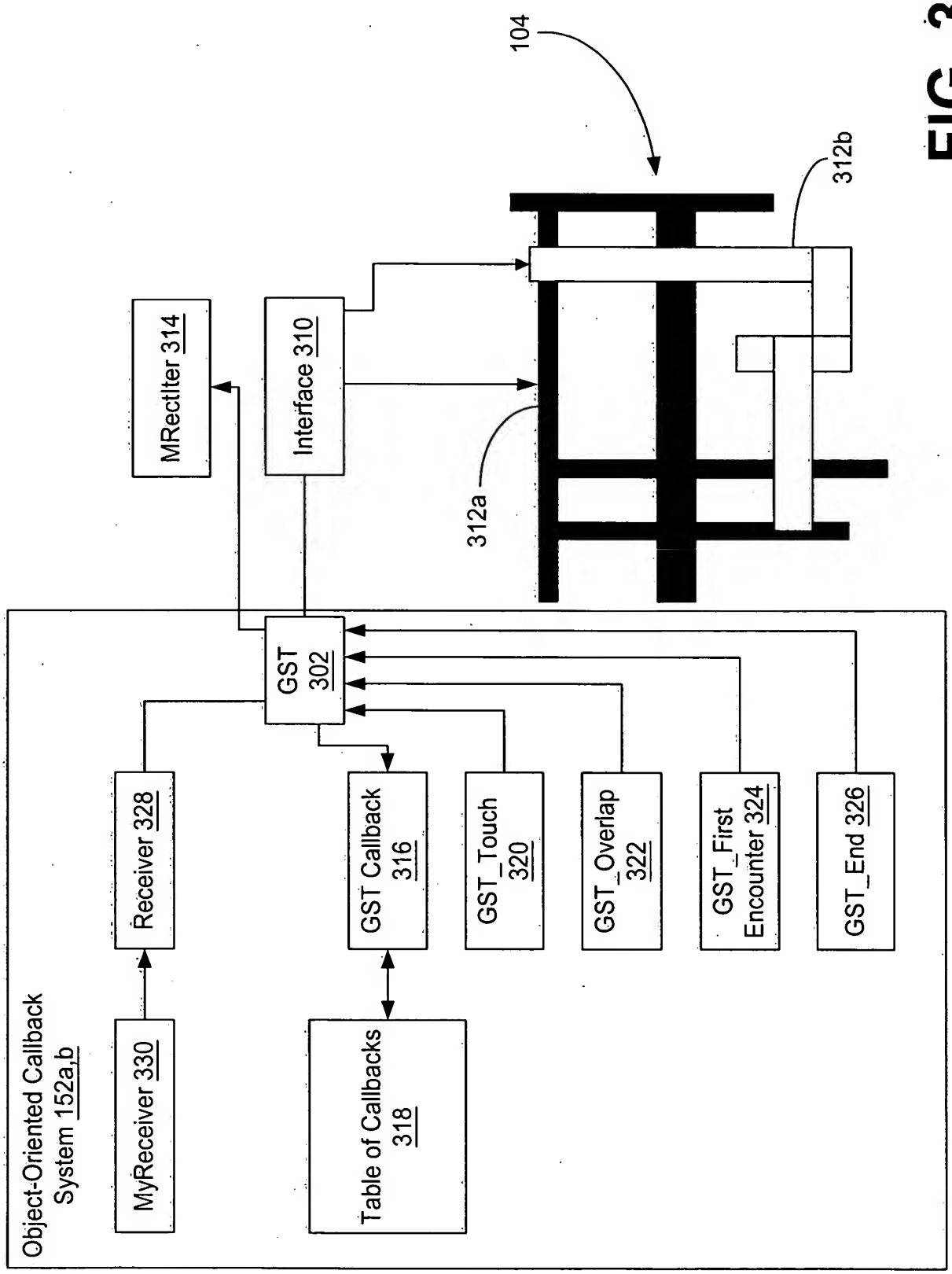

104
312b
312a
MRectIter 314
Interface 310
GST 302
Receiver 328
GST Callback 316
GST_Touch 320
GST_Overlap 322
GST_First Encounter 324
GST_End 326
MyReceiver 330
Table of Callbacks 318
Object-Oriented Callback System 152a,b

FIG. 3

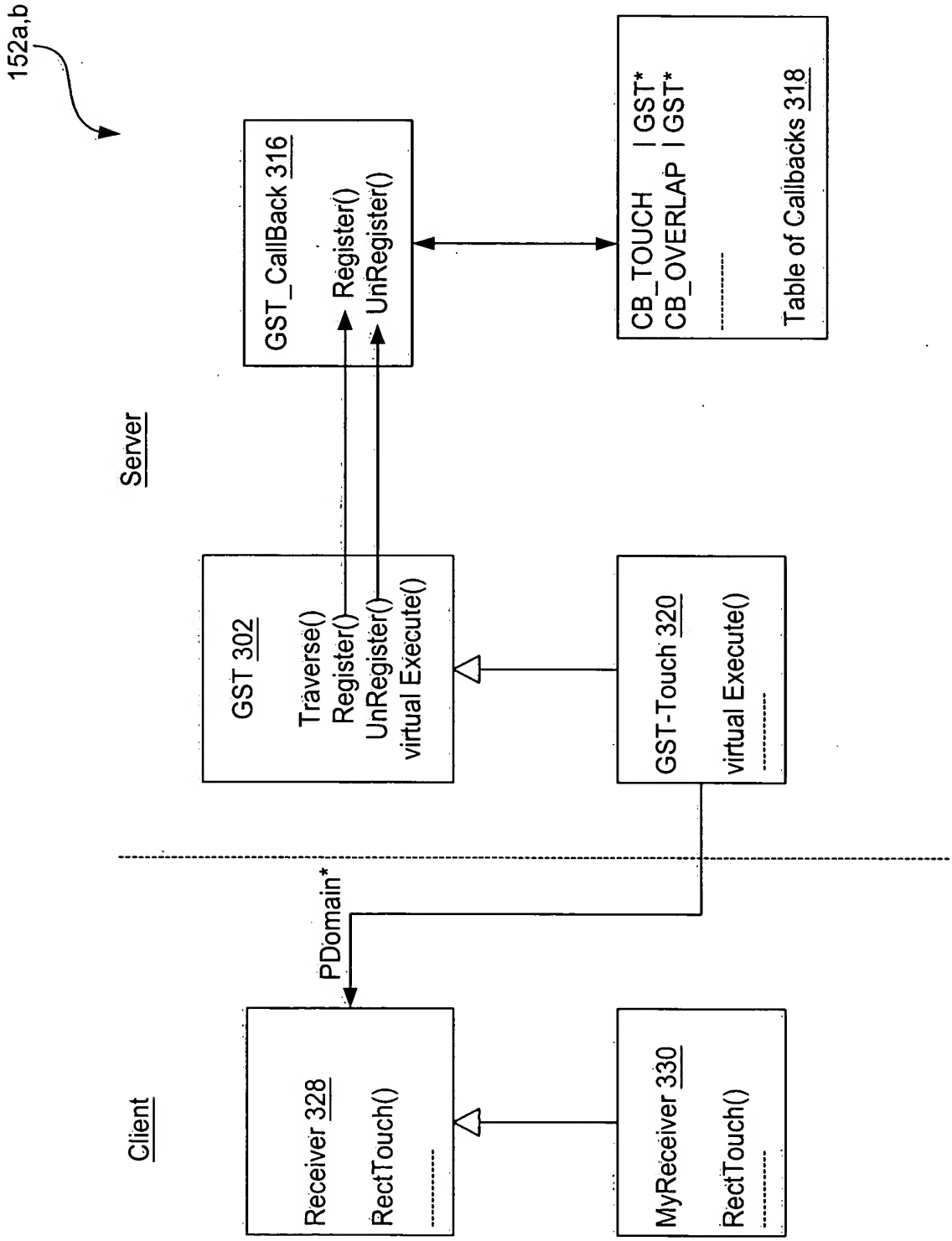

152a,b
Server
GST_CallBack 316
Register()
UnRegister()
CB_TOUCH | GST*
CB_OVERLAP | GST*
Table of Callbacks 318
GST 302
Traverse()
Register()
UnRegister()
virtual Execute()
GST-Touch 320
virtual Execute()
PDomain*
Client
Receiver 328
RectTouch()
MyReceiver 330
RectTouch()

FIG. 4A

Client Perspective

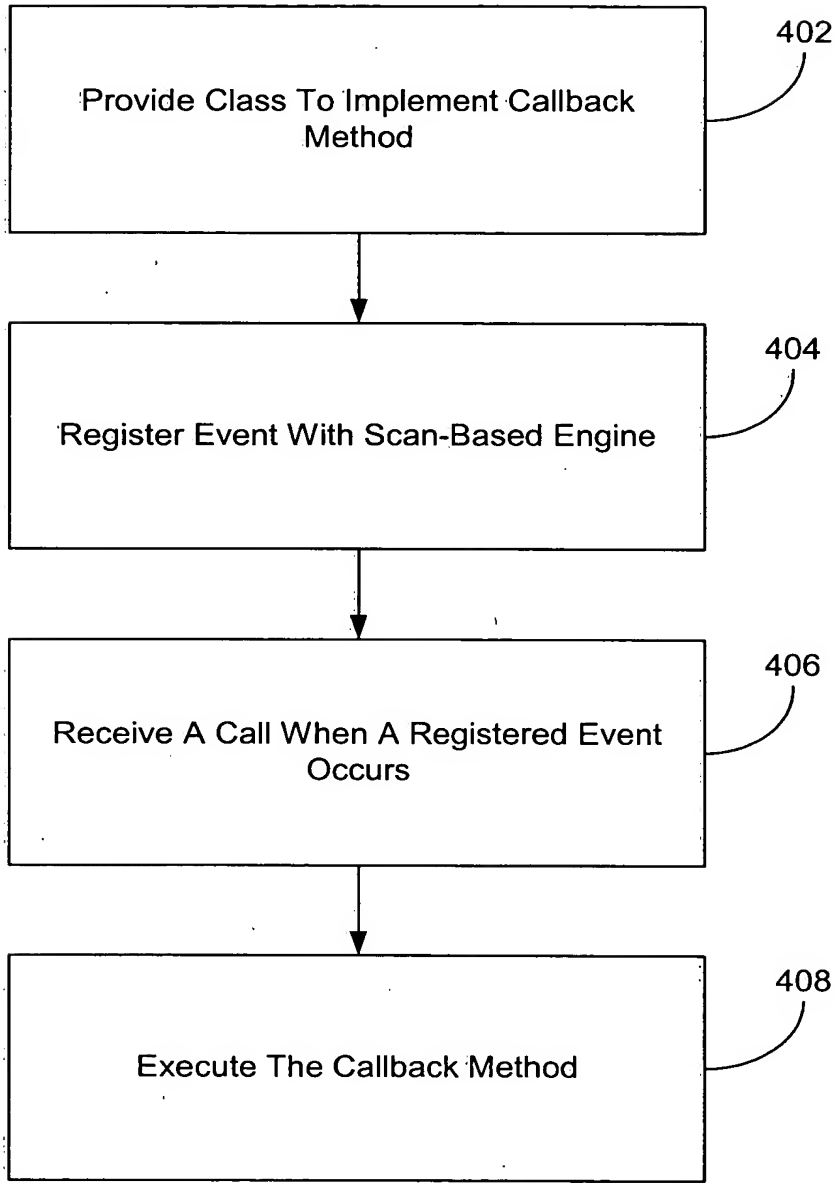


FIG. 4B

Server Perspective

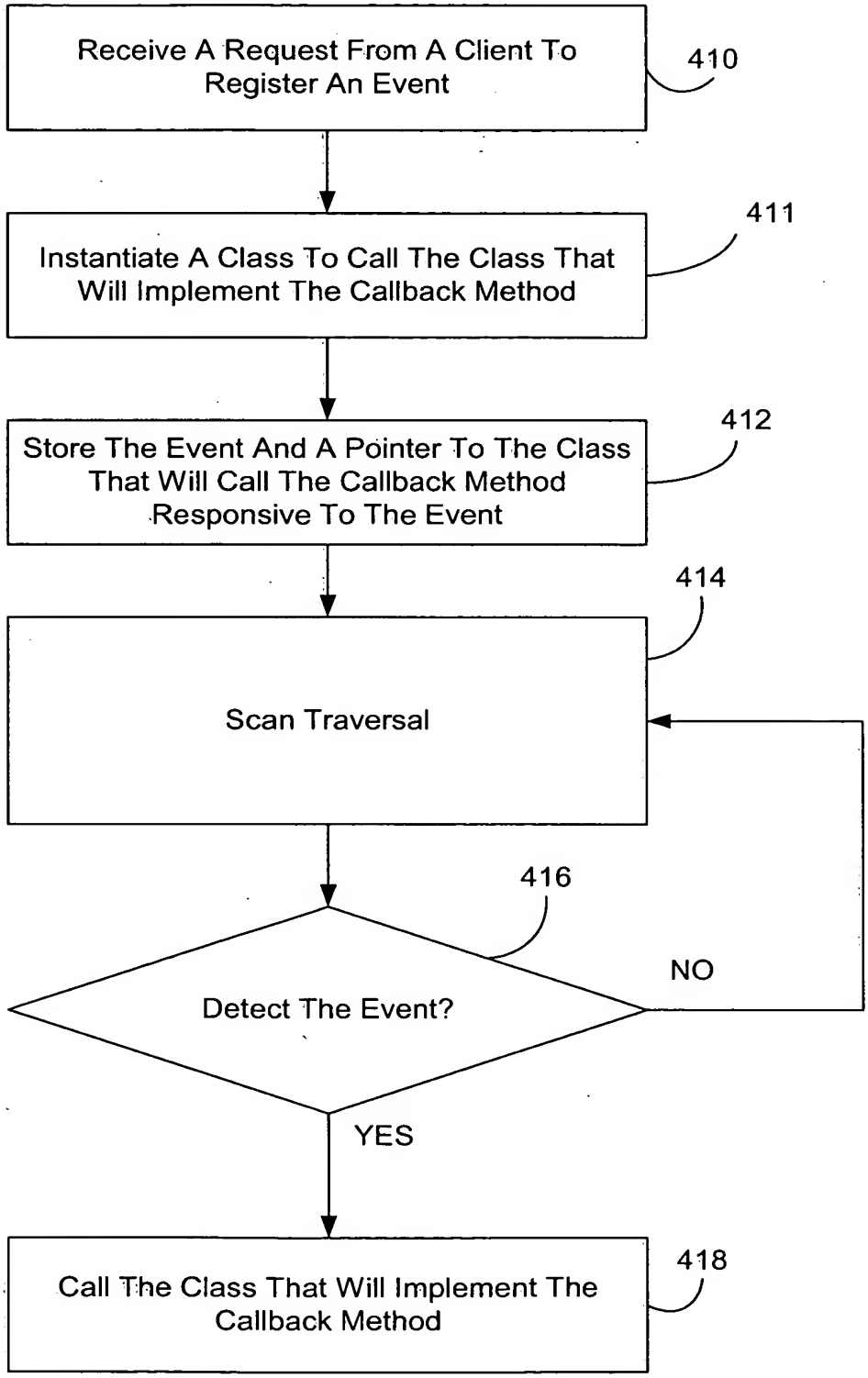


FIG. 4C

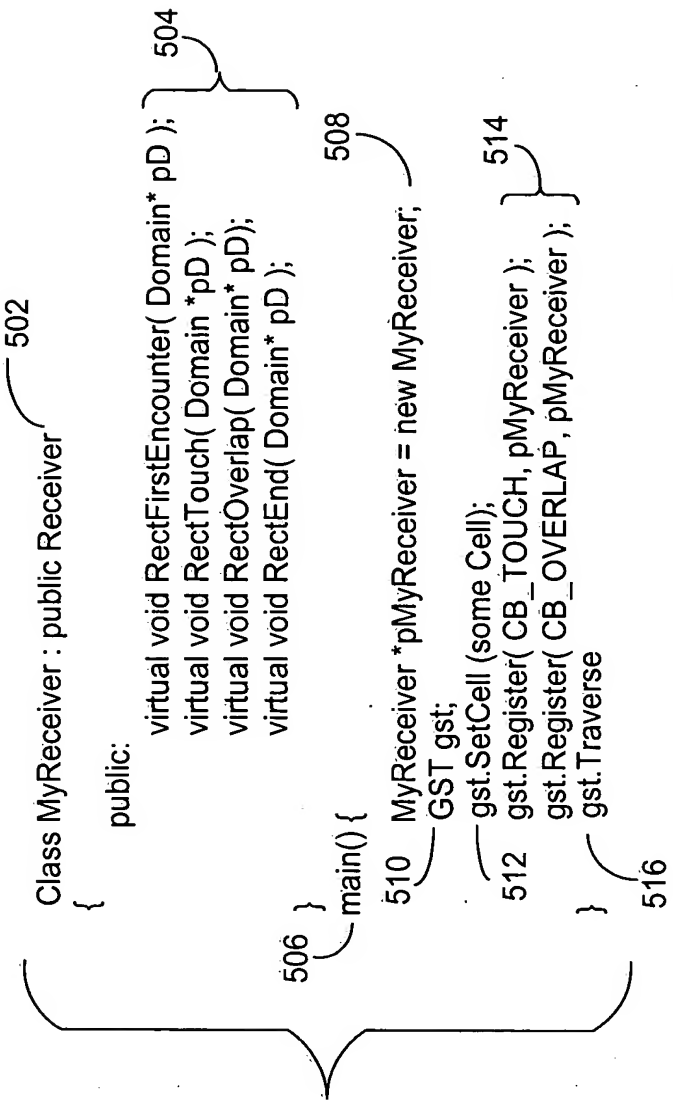

Class MyReceiver : public Receiver
{
public:
virtual void RectFirstEncounter(Domain* pD);
virtual void RectTouch(Domain *pD);
virtual void RectOverlap(Domain* pD);
virtual void RectEnd(Domain* pD);
}
main() {
MyReceiver *pMyReceiver = new MyReceiver;
GST gst;
gst.SetCell (some Cell);
gst.Register(CB_TOUCH, pMyReceiver);
gst.Register(CB_OVERLAP, pMyReceiver);
gst.Traverse
}
502
504
506
508
510
512
514
516

FIG. 5A

Void MyReceiver::RectTouch(Domain* pD)
{
if (pD)
{
a = pD ->GetRect();
b = pD ->GetLayerName();
}
}
518
520

FIG. 5B

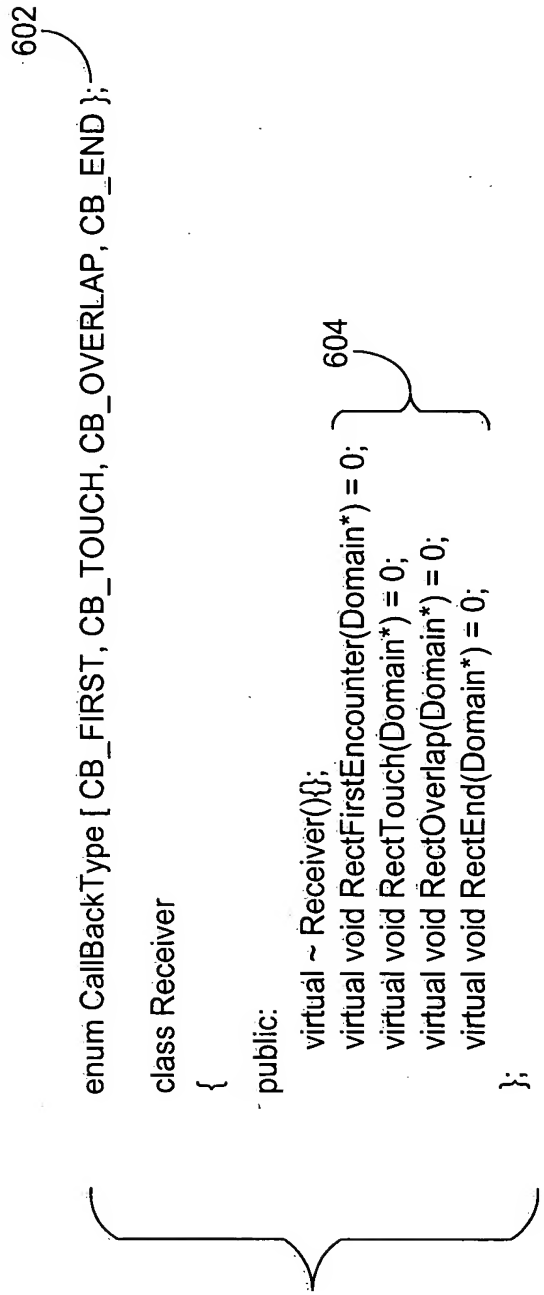

602
enum CallBackType [CB_FIRST, CB_TOUCH, CB_OVERLAP, CB_END };
class Receiver
{
public:
virtual ~ Receiver(){};
virtual void RectFirstEncounter(Domain*) = 0;
virtual void RectTouch(Domain*) = 0;
virtual void RectOverlap(Domain*) = 0;
virtual void RectEnd(Domain*) = 0;
};
604

FIG. 6A

class GST
{
public;
GST();
~GST();
bool Traverse();
.....
606
void Register(CallBackType, Receiver*);
void UnRegister(CallBackType);
}
608

FIG. 6B

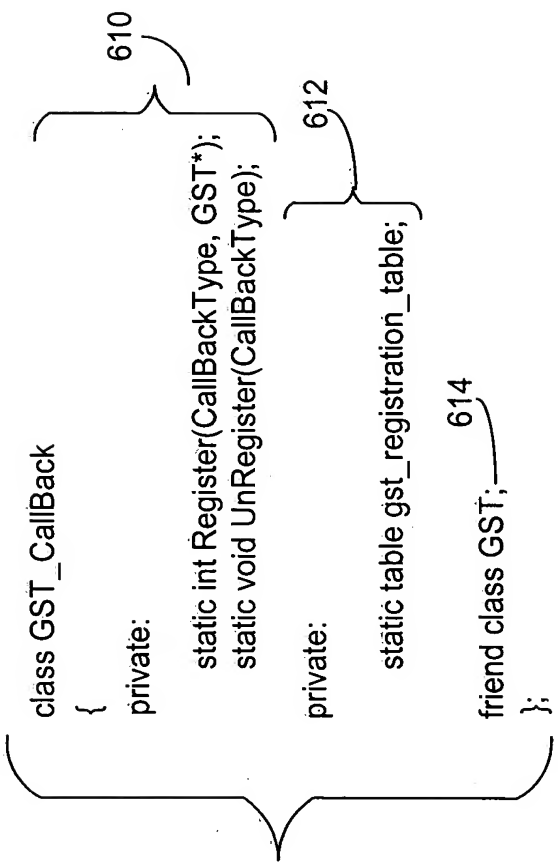

class GST_CallBack
{
private:
static int Register(CallBackType, GST*);
static void UnRegister(CallBackType);
610
private:
static table gst_registration_table;
612
friend class GST;
614
};

FIG. 6C

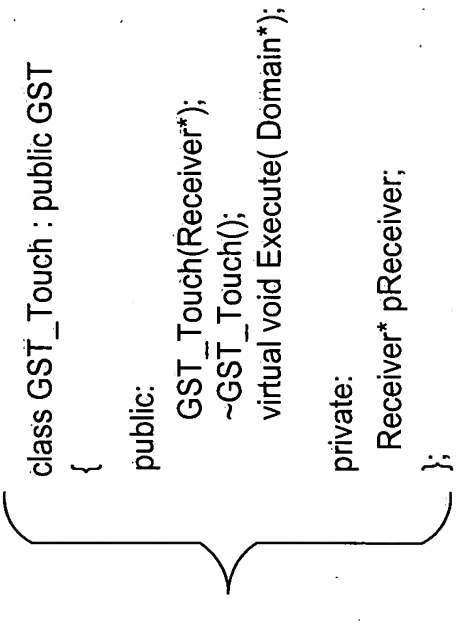

class GST_Touch : public GST
{
public:
GST_Touch(Receiver*);
~GST_Touch();
virtual void Execute(Domain*);
private:
Receiver* pReceiver;
};

FIG. 6D

FIG. 6E

```
int GST_CallBack::Register( CallBackType aCB, GST*, pGST )
{
    if ( gst_registration_table.insert(make_pair(aCB, pGST)).second )
        return 1;
    else
     {
        cout<<"GST_CallBack::Register Error: Key already exists" << endl;
        return -1;
     }
}
```

FIG. 6F

```
void GST_CallBack::UnRegister(CallGackType aCB )
{
  gst_registration_table.erase( aCB );
}
```

FIG. 6G

```
void GST_Touch::Execute( Domain* pD )
{
  pReceiver ->RectTouch( pD );
}
```

FIG. 6H

```
GST::Traverse()
{
 ----------
 pTouch = GST_CallBack::gst_registration_table.find( CB_TOUCH)) ->second;
 ----------
 pTouch ->Execute( pDomain );
}
```

Client Perspective

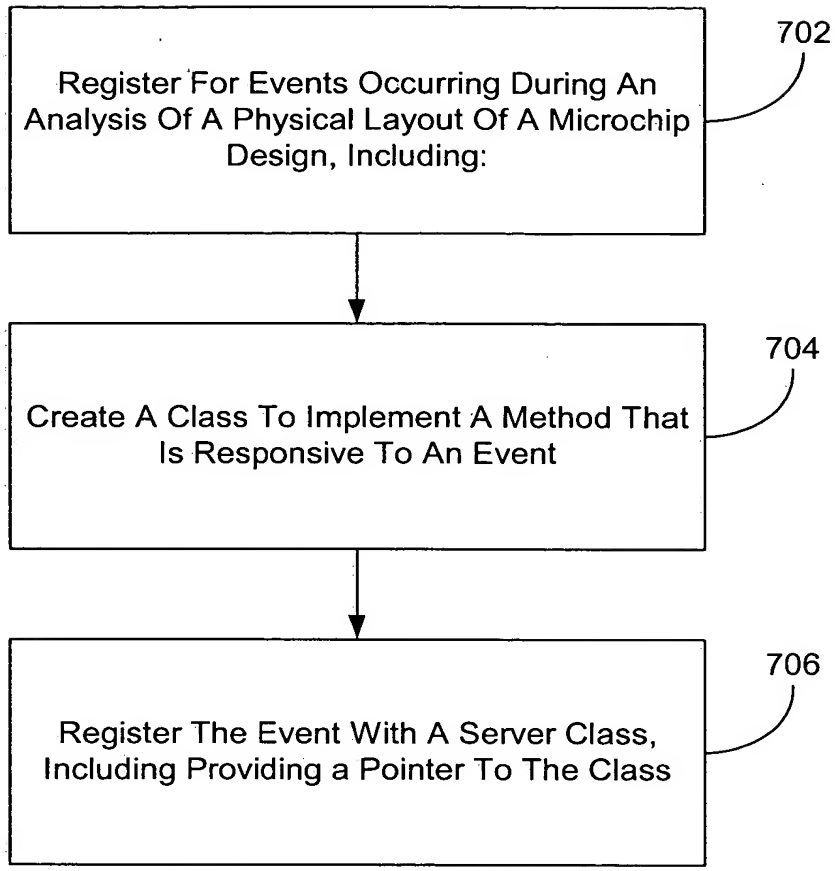


FIG. 7

13/15

Server Perspective

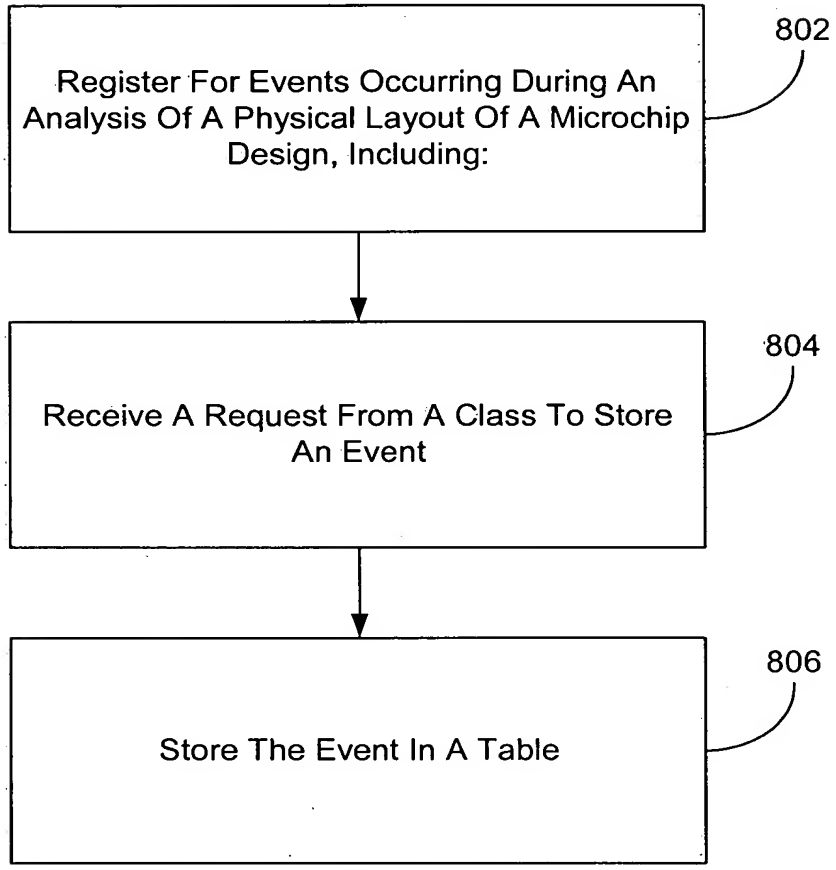


FIG. 8

Client Perspective

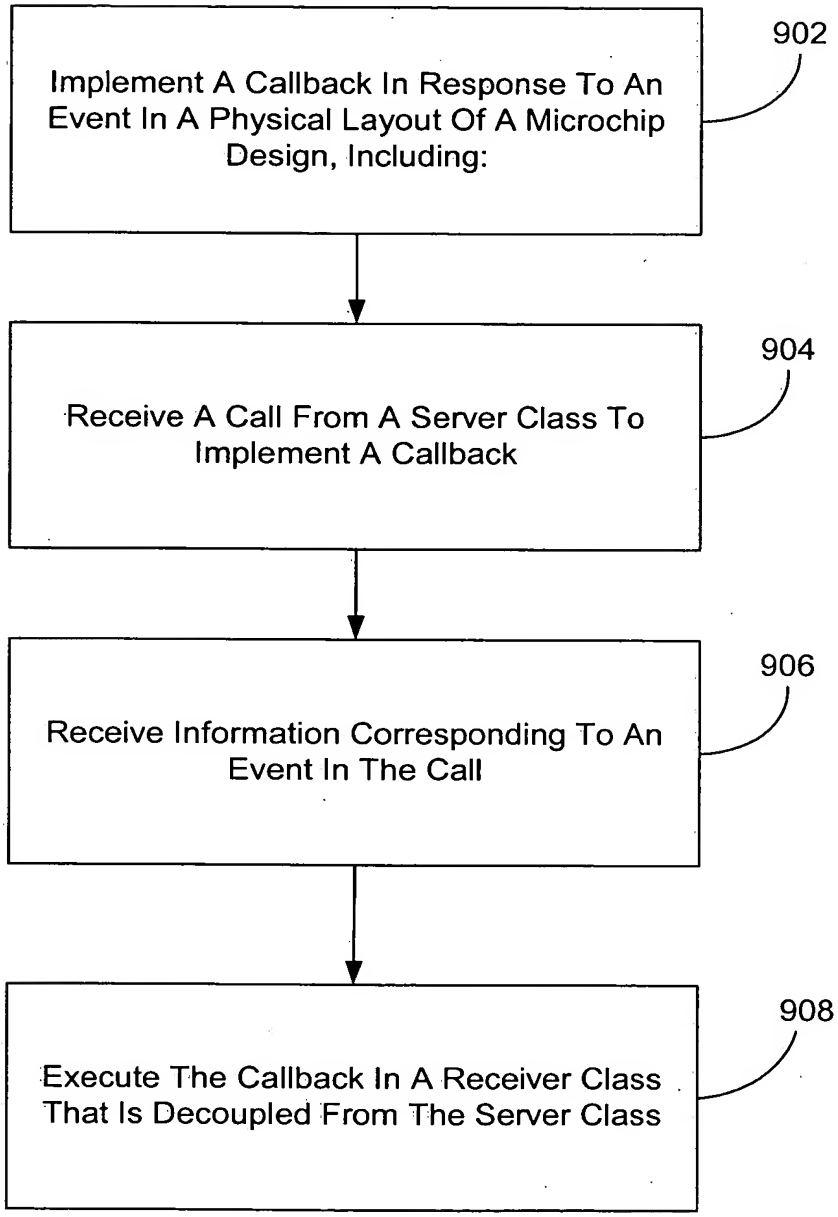


FIG. 9

15/15

Server Perspective

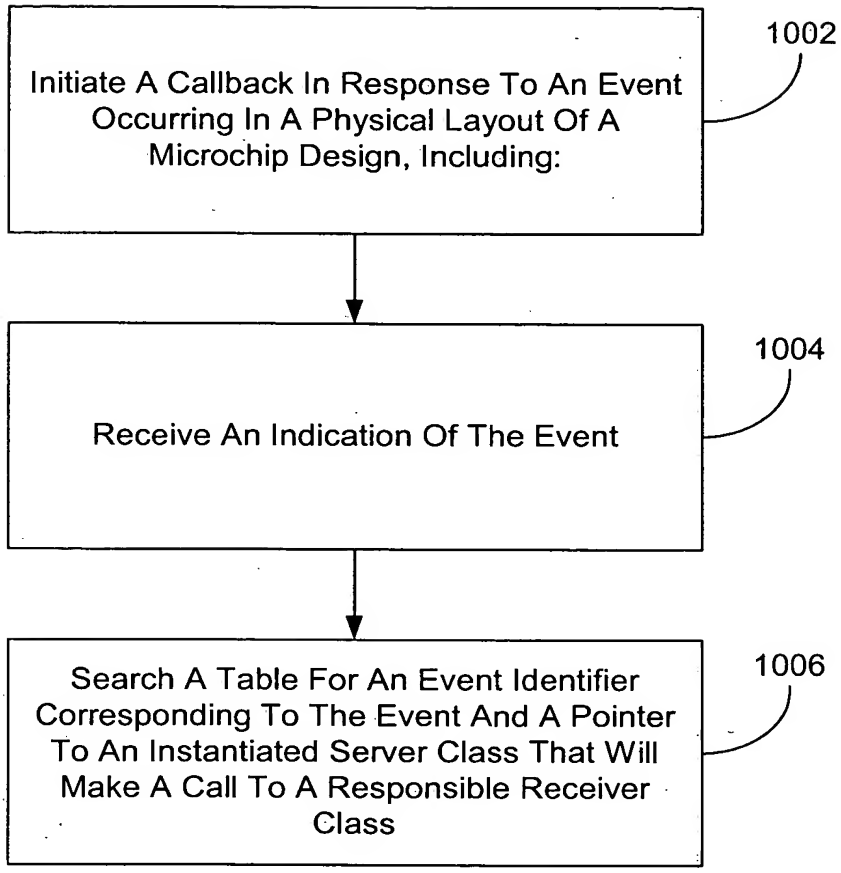


FIG. 10